

ALGORITMOS Y ESTRUCTURAS DE DATOS III - 1^{er} Recuperatorio
Fecha de examen: 28-NOV-2022

Notas:	Nº Orden	Apellido y nombre		L.U.	# hojas ¹
	Ej1	Ej2	Ej3	Ej4	Final

Aclaraciones: El parcial se aprueba con 2 (dos) ejercicios aprobados. Cada hoja debe estar numerada y debe tener el número de orden y L.U. El parcial dura 4 horas y es a libro abierto.

1) El mejor amigo de Tuki, Pipo, también quiere aprender a contar cadenas. Como Tuki es un maestro de esta técnica, ahora le da ejercicios a su amigo para que aprenda y practique. Puntualmente, le propuso el siguiente desafío: contar cuántas cadenas de números naturales $a_1 \dots a_n$ existen tales que $0 \leq a_i \leq c$ para todo i , y exactamente k números de la cadena son primos. Como Pipo no creyó que este problema fuera lo suficientemente interesante, Tuki agregó la condición de que la cadena no puede tener m números consecutivos que no sean primos. Para facilitar su trabajo, le alcanzó una función ESPRIMO que permite calcular, dado un número x , si x es primo en $O(1)$.

a) Definir en forma recursiva la función $f_{c,m} : \mathbb{N}^3 \rightarrow \mathbb{N}$ donde $f_{c,m}(i, k, t)$ calcula cuántas cadenas $a_1 \dots a_i$ existen tales que $0 \leq a_j \leq c$ para todo j , la cantidad de números primos entre los a_j es exactamente k , nunca ocurre que m números consecutivos no son primos, y, si $t \leq i$, **entonces hay un a_j con $1 \leq j \leq t$ tal que a_j es primo**. Señalar qué llamado debe hacer Pipo para resolver el problema.

Ayuda: Utilice el tercer parámetro para asegurar que no haya m números consecutivos que no sean primos.

b) Demostrar que $f_{c,m}$ tiene la propiedad de superposición de subproblemas.

c) Definir un algoritmo *top-down* para calcular $f_{c,m}(i, k, t)$ indicando claramente las estructuras de datos utilizadas y la complejidad resultante.

d) Escribir el (pseudo-)código del algoritmo top-down resultante.

Complejidad: la complejidad temporal del algoritmo resultante para calcular $f_{c,m}(i, k, t)$ debe ser $O(c + i \min\{i, k\} \min\{i, m\})$.

2) Un *modelo de intervalos* es una secuencia $\mathcal{I} = [s_1, t_1], \dots, [s_n, t_n]$ de intervalos cerrados tales que $0 < s_1 < s_2 < \dots < s_n$. Decimos que un conjunto de puntos $P \subseteq \mathbb{R}$ es un *transversal* de \mathcal{I} cuando $P \cap [s_i, t_i] \neq \emptyset$ para todo $1 \leq i \leq n$.

a) Proponer un algoritmo goloso que, dado un modelo de intervalos \mathcal{I} , encuentre el transversal mínimo de \mathcal{I} .

Ayuda: considere cuál es la mejor forma de colocar el punto $p \in P$ que pertenece a $[s_n, t_n]$. Luego, genere P de mayor a menor.

b) Demostrar que el algoritmo propuesto es correcto.

Ayuda: sugerimos demostrar por inducción una propiedad con el mismo espíritu que aquella del trabajo práctico pero adaptada al problema en cuestión. Tener en cuenta que si P se genera de mayor a menor, entonces la inducción debe ser en el mismo sentido.

Complejidad: la complejidad temporal del algoritmo que encuentra el *transversal* P debe ser $O(n)$.

¹Incluyendo esta hoja.

- 3) Una empresa de comunicaciones tiene un conjunto de nodos proveedores. Existen conexiones entre los nodos a distintos clientes formando una red. Cada conexión tiene una capacidad entera asociada que representa su ancho de banda. La empresa modela esto utilizando un grafo pesado G .

El *ancho de banda proveedor* es el máximo k tal que existe un nodo proveedor en cada componente conexa de G_k , donde G_k es el subgrafo generador de G que se obtiene de eliminar las aristas de peso menor a k .

- a) Dado un grafo $G = (V_{\text{clientes}} \cup V_{\text{proveedores}}, E)$. Proponer un algoritmo eficiente para determinar el *ancho de banda proveedor* de un grafo G .
- b) Supongamos que la empresa puede hacer que **un único** cliente de la red se transforme en nodo proveedor. Dar un algoritmo para encontrar algún cliente que al transformarlo en nodo proveedor, mejore el *ancho de banda proveedor* de la red. Si no es posible, indicarlo; y si hay más de un cliente que se pueda elegir, devolver a todos ellos.

Complejidad: la complejidad de ambos algoritmos debe ser $O(\min\{m \log(n), n^2\})$.

- 4) Dado un grafo G , la *excentricidad* de un vértice $v \in V(G)$ se define como $e(v) = \max_{u \in V(G)} d(v, u)$, siendo d la función de distancia. El *centro* de G es el conjunto $C_T \subseteq V(G)$ tal que $v \in C_T \Leftrightarrow e(v) = \min_{u \in V(G)} e(u)$ (Es decir, los vértices cuya distancia máxima hacia cualquier otro vértice es mínima). Sea T un árbol y $P = v_0, v_1, \dots, v_k$ un camino de longitud máxima en T .

- a) Sea $p = \lceil k/2 \rceil$. Probar que si k es par, entonces $e(v_p) = p$, y si k es impar, $e(v_{p-1}) = e(v_p) = p$.
Ayuda: Considere el efecto de tendrías sobre el camino P que $e(v_p)$ (o $e(v_{p-1})$ en el caso k impar) fuera mayor a p .
- b) Pruebe que si $w \notin P$, entonces $e(w) > e(v_p)$. Concluya que si k es par, entonces v_p es el único centro del grafo, mientras que si k es impar, entonces v_p y v_{p-1} lo son.
Ayuda: Suponga que w posee excentricidad menor o igual a v_p , y tenga en cuenta los caminos $w \rightsquigarrow v_0$ y $w \rightsquigarrow v_k$, considerando sus longitudes y puntos de intersección con P .
- c) Dar un algoritmo que en $O(n)$ determine el/los centros de un árbol T . Puede asumir que cuenta con un algoritmo que obtiene un camino máximo de T en $O(n)$ tiempo².

²Ejercicio 4 del primer parcial